



# Markov chain Monte Carlo for the reconstruction of lineage trees from single-cell DNA data

## Markov chain Monte Carlo för rekonstruktion av stamträd från encells DNA-data

BALCÁZAR CASTELL, IRIS

Double degree Project in Computer Science and Mathematics
Date: July 2, 2019
Supervisor: Jens Lagergren, Jeanette Hellgren Kotaleski and Marta Casanelles
Examiner: Örjan Ekeberg
School of Electrical Engineering and Computer Science
Host company: Science for Life Laboratory

# Abstract

The purpose of this study is to infer evolutionary trees through the Markov chain Monte Carlo algorithm (MCMC) [1] based on whole-genome single-cell DNA sequencing data.

By using MCMC we obtain likely tree structure samples according to the cells' somatic point mutations in our data. This probabilistic framework takes into consideration the errors caused by the current technology such as amplification errors, sequencing errors and allelic dropouts.

We investigated whether using this technique is reasonable given this biological scope. Most of the results give interesting conclusions that improve the previous results on the same Site Pair Model [2] and therefore we conclude that using MCMC is reasonable. Though, since the model is based on probabilities and the algorithm randomizes decisions the best results are not always guaranteed. One needs to be aware that a decent amount of data in the data set is an important requisite to predict accurate tree structures. Furthermore, the computational time for this process is significantly high and can not be computed on regular laptops for large and realistic data sets. This is acceptable since for this type of research speed is not a strict requirement: it is worth waiting more for a given execution if the obtained results are more interesting or more accurate.

Finally, we propose some further improvements for this strategy that could potentially obtain even better results in terms of accuracy and speed.

# Sammanfattning

Syftet med denna studie är att utgå från evolutionära träd genom Markovkedjan Monte Carlo-algoritmen (MCMC) [1] baserat på DNA-sekvenserings data med heltgenet.

Genom att använda MCMC får vi sannolikt trädstrukturprover enligt cellens somatiska punktmutationer i våra data. Detta probabilistiska ramverk tar hänsyn till de fel som orsakas av den nuvarande teknologin, såsom förstärkningsfel, sekvenseringsfel och allelavfall.

Vi undersökte om användningen av denna teknik är rimlig med tanke på detta biologiska omfång. De flesta av resultaten ger intressanta slutsatser som förbättrar tidigare resultat på samma Site Pair Model [2] och därför sluts vi att MCMC är rimligt. Trots att modellen är baserad på sannolikheter och algoritmen randomiserar beslut är de bästa resultaten inte alltid garanterade. Man måste vara medveten om att en anständig mängd data i datasatsen är en viktig förutsättning för att förutsäga exakta trädstrukturer. Dessutom är beräkningstiden för denna process betydligt hög och kan inte beräknas på vanliga bärbara datorer för stora och realistiska dataset. Detta är acceptabelt eftersom för denna typ av forskningshastighet inte är ett strikt krav: det är värt att vänta mer på ett visst utförande om de erhållna resultaten är mer intressanta eller mer exakta.

Slutligen föreslår vi några ytterligare förbättringar för denna strategi som potentiellt kan få ännu bättre resultat när det gäller noggrannhet och snabbhet.

***Keywords—*** Markov chain Monte Carlo, MCMC, Bayesian statistic, Phylogeny, Genome modeling.

# Contents

# Chapter 1

# Introduction

Understanding the natural behavior and development of human beings has been the center of interest of many scientists and researchers since the beginnings of science. Throughout years of experimentation and research, some of the mysteries of human health have been understood increasing our medical skills continuously.

Nowadays, the evolution of technology has allowed us to get closer and closer to the key of our identity, our DNA. Being able to read the nucleotide sequences that form our DNA in each cell (see 2.4. Notation for brief definition), driving to the opportunity to analyze how it affects to the cell. Are two cells from the heart more similar to each other than a skin cell? Are closer cells of the brain more similar than the ones further away? Research over the effect of DNA sequencing mutations is a big step for understanding diseases caused by them, among other cancer for example, and this knowledge could allow us to develop prevention or treatments for it.

A big research area in generics is phylogeny [3], which consists in defining evolutionary trees to analyze the differences between species, or cells in our case, in a more structured and statistical way. Each leaf of the tree represents one of the analyzed individuals and the branches represent the closeness between them.

Finding the most likely tree for a given set of data requires interesting methods and many different strategies have been used. In this thesis, not all of them will be covered. Still, an insight in distance methods [3] and Bayesian statistic [4] is needed since they will be powerful tools for the development of the study. The first one refers to strategies that allow us to generate evolutionary trees approximations at not to high computational cost, even with big amounts of input data. It does so by clustering the most similar species based on the distance between them, that is, based on how similar to each other they are. The second is the basic statistical concept used in the Markov chain Monte Carlo algorithm [4, 1], from now on MCMC, which we aim to use to provide samples for cell lineage trees from the given data to extract statistical conclusions about the similarities among cells. This will give some differences towards the previous strategy and show whether this method works better or not.

For this project, the sources for the DNA sequences are Karolinska Institute laboratory's experiments in DNA sequencing and a public data set provided by the National Center for Biotechnology Information at the US National Library of Medicine [5]. As the reader will see further on and due to the high computational cost of such large data sets the testing for the code will be mainly done through smaller synthetic generated data.

## 1.1 Research Question

In this thesis, we will investigate over the data in cells DNA provided by single-cell genome sequencing [6] for healthy tissue to answer to the following question: Can accurate cell lineage trees be provided by using MCMC algorithm?

# Chapter 2

# Background

There are at least two important reasons why this research area of phylogeny has grown in the last years. On one hand, the increased amount of cancer detection and mortality. Usually, once a disease starts affecting a larger amount of people, more effort is put to find cures and preventive actions. Among others, cancer is characterized by an accumulation of somatic mutations in a group of cells [7]. This means that the DNA sequence of the cell is altered after conception.

On the other hand, the previously mentioned single-cell genome sequencing technique that allows us to obtain the DNA sequence from a cell. Instead of reading from a group of cells that may differ to each other in some nucleotides of the sequence (some particles of the DNA), we can take the information one by one. Due to the fact that some alterations can happen in the cells reproduction and due to somatic mutations close cells may also be different and therefore it is more accurate to differentiate their reads.

It has been proved that there is intra-tumor heterogeneity in mutations, which means that the cells inside a given tumor harbor the same mutation types, whereas from tumor to tumor they are not necessarily equal [8]. The key would be to conclude which mutations are causing the abnormal growth of the cells to understand the disease fully. Since this issue is complex and slow, understanding the impact of mutations in healthy tissue first is very interesting. Dealing with healthy tissue is more reliable and useful to compare later to the characteristics of diseased tissue. In diseased tissue, not only the mutations alter the sequence but also whole segments can be missing, resulting in a very complex problem to solve.

Please note that a biological basic notion regarding cellular DNA structure will be assumed throughout this project. Nevertheless, to make it easier to understand for those readers without biological background the most general structure will be introduced here.

Inside the nucleus of every cell the genetic information is defined, through the chromosomes. They are formed by two chromatids, that contain same DNA informa-

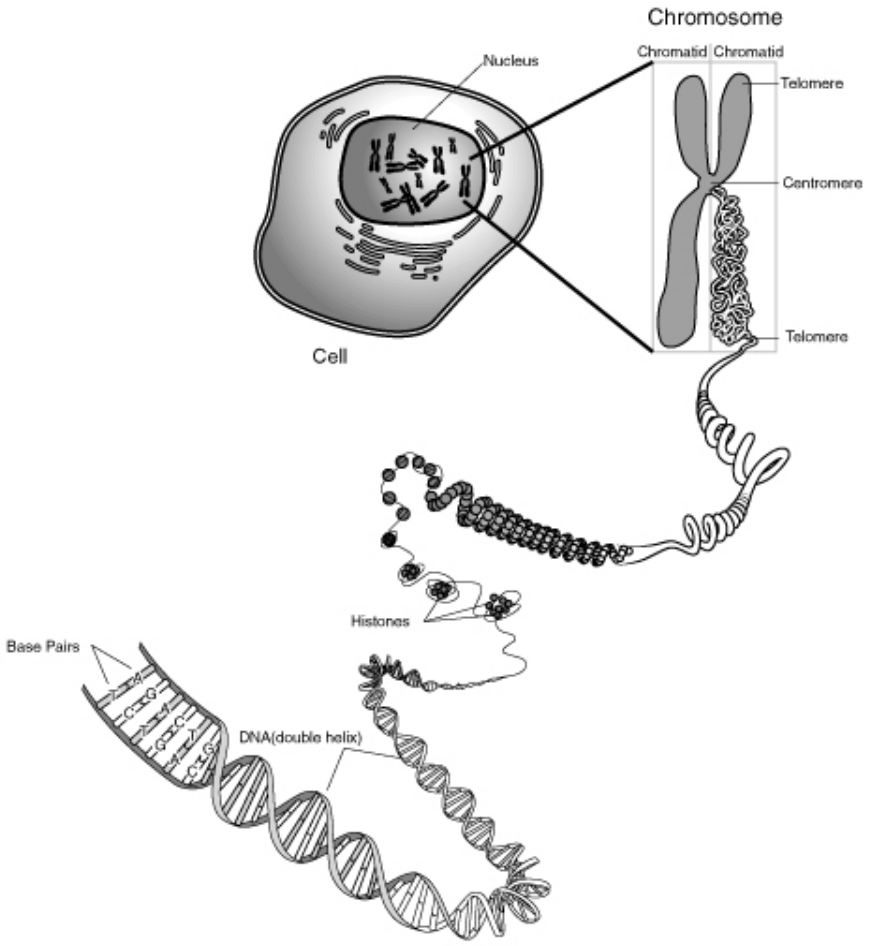



Figure 2.1: Cell genetic structure

tion in the DNA sequence helix. In this thesis, it is important to know that in this the helix there are base pairs (nucleotide pairs) generating a specific sequence. Each position of the sequence will be referred to as *site* and sites with somatic mutations are those with a replacement of the nucleotide by another nucleotide.

## 2.1 Technical background

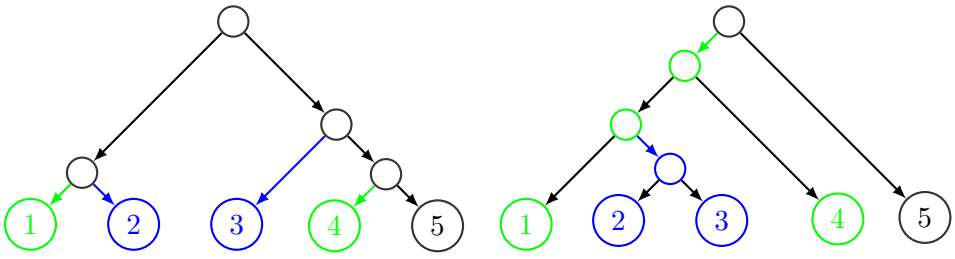
In this section, we will focus on the relevant aspects of this area for the goals of the project regarding technical methods.

### 2.1.1 Parsimony methods

When we consider evolutionary trees and aim to determine the difference among different species though the DNA sequence, a simple way to do it is by counting the number of differences between them. The higher the number of positions that differ, the further away in the evolutionary tree this species will be. The simplicity of this idea, having the least amount of net evolution, is a good starting point to understand the issues raised in this project [3].

Let us introduce this concept with a simplified example. Consider five cells and consider mutations in the cells for a single site i.e. for a given position. The mutations will be represented in the schemes as a change in the color and the location where the mutation happened will be indicated by coloring the correspondent arrow. The tree structure is unknown and we aim to find the best one. In this example, the bulk cell will be represented as black, i.e. the non-mutated root cell. The data shows cells 1 and 4 mutated to green, cells 2 and 3 mutated to blue and cell 5 still equal to bulk.

There are many possibilities when proposing a lineage tree. Among all of them, the most parsimonious evolutionary trees are those with the least amount of changes of state.




Proposed tree 1 Proposed tree 2

Schema 2.1: Phylogenies for simplified data case

If we consider the *Proposed tree 1* as the phylogeny for our data we can easily see that at least four changes of state are required, indicated in the schemes with the colored arrows. Starting with black, the only way to obtain this data is by having a color mutation in at least 4 branches. Taking the *Proposed tree 2* as the tree structure for the data we can see that the necessary changes are only 2 since each mutation effects to all its children nodes. This makes it the most parsimonious tree structure.

There are two main difficulties when searching for the most parsimonious trees:

1. we need to be able to determine the minimum amount of changes that happen given a proposed tree and

2. we need to be able to construct all possible tree structures.

In real cases where the number of cells and the number of sites per cell is realistically high[1], this becomes computationally a significant issue.

This brings the researchers to figure out alternative methods to determine evolutionary trees through cell DNA data, such as Markov chain Monte Carlo (MCMC) algorithms.

### 2.1.2 Maximum likelihood methods for MCMC

The method used for our problem has also been worked on previously. MCMC algorithm consists in sampling linked evolutionary trees by modifying the current state to the next and aiming to obtain the sample that is closest to the real phylogeny of the considered cells. The specification for the used application will be developed further on.

At this point, it is important to understand how to compute how likely a tree is given the cells data. The most standard framework for it is the maximum likelihood, a statistical method drives, as parsimony methods, to the least amount of net evolution. The application of maximum likelihood for phylogenies was introduced by Edwards and Cavalli-Sforza in 1964 for gene frequency data [9]. In 1971 Jerzy Neyman used it for molecular sequences [10], a work that was developed by other statisticians during the seventies.

The likelihood works in a general scope as follows: given some data $D$, the likelihood of a hypothesis $H$ is proportional to the probability of obtaining $D$ assuming that $H$ is true. So, the likelihood is proportional to $P(H|D)$, though it is not a probability on its own. The hypothesis may vary, but the data is always the same, giving information about what parameters in the hypothesis make the obtained data more likely. In this study, the data $D$ are the reads of genome sequences of single cells obtained in the laboratory and the hypothesis is the full evolutionary tree. A higher likelihood for the tree will mean that the probability of obtaining the treated data is

[1] For human beings the total amount of sites can be over hundreds of thousands.

also higher for this tree structure. So, we can keep modifying the trees to increase this value and therefore to get closer to the real evolutionary tree. This is the idea of what the Markov chain Monte Carlo does.

## 2.2 Previous work

Currently, some programming languages such as Python or R have already implemented this algorithm due to its relevance and utility. Though, the implementation is subjected to the model, and every model its limitations.

It is especially important to be aware of the possible errors in the DNA processing (duplicating DNA fragments of the cell to have enough samples to be read, called genome amplification) and sequencing (the readings of the nucleotides themselves causing false-positives). If the researcher considers these errors to be mutations, it can lead to an incorrect interpretation of the results. Furthermore, in both processes, the result may be missing segments of the DNA sequence since the amplification is not homogeneous among the whole DNA and the reading coverage is not total. We will explain these issues further on and define how our model, differently than other models, takes this aspects into consideration.

### 2.2.1 At Science for Life Laboratory

This project is a further development of one of the research groups at Science for Life Laboratory[2]. Therefore their model to identify somatic mutations will be used: the site pair model presented in *SCuPhr: A Probabilistic Framework for Cell Lineage Tree Reconstruction* by Hazal Koptagel, Seong-Hwan Jun and Jens Lagergren [3] [2]. Mainly, it consists in identifying two sites on the DNA sequence where one site is homozygous (same nucleotide for both chromatids) and the other is heterozygous (different nucleotides), see 2.4. Notation for a brief description. The reads for each cell in a given pair of sites will consist of two pairs of nucleotides (the pair belonging to the two chosen sites in each chromatid). Based on this reads we aim to establish an evolutionary tree that represents the possible mutations taken place in these sites, assigning cells with the same mutation in these sites to closer branches of the tree. Doing this for a big amount of pairs in the sequence we aim to find a tree that agrees with as many as possible sites. It should not require to satisfy all the pairs of sites due to the possible errors in the data sequencing that we mentioned before.

Using this method instead of the traditional, allows us to increase the accuracy since we are taking into account a larger subset of the given data. Not only homozy-

[2]More information regarding the Science for Life Laboratory can be found in the website www.scilifelab.se.

[3]All three from Science for Life Laboratory, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

gous positions are visited but also heterozygous. Note that with only one site it was not possible to conclude full information regarding the mutations. Imagine that both chromatids have nucleotide $A$ in the given site of the bulk (non-mutated cell). When reading $C$, for example, we would not know which chromatid is mutated. When adding the second site, we can identify the position of these mutations. Let's say the first chromatid has $A$, $T$ and the second chromatid has $A$, $C$ in the given pair of sites. Finding $C, T$ in the reads would determine that the mutation comes from the first chromatid and $C, C$ from the second.

At SciLifeLab this model is investigated and tested. The used technique until now is a distance method that generates an approximation of the phylogeny. In specific, the neighbour-joining (NJ) algorithm by Saitou and Nei, 1987.

## 2.3 Assumptions

We will make the following assumptions:

**A1:** *We assume infinite sites model.* This implies that no site is hit with mutation twice, i.e., the probability that a site is hit with two mutations is negligible. Assuming this is reasonable due to the large amount of data that is considered.

**A2:** *The nucleotides at different sites evolve independently.* This is a standard assumption in the analysis of DNA-seq data [11].

**A3:** *The tissue has no account of copy number variations.* This assumption ensures that somatic mutations provide sufficient evidence for constructing the cell lineage tree.

## 2.4 Notation

Throughout this project the following notation will be used.

**Nucleotides:** The nucleotides are organic molecules that form the DNA, composing the two chromatids of the human genes in helix shape (generally $XY$ or $XX$). The four nucleotide bases are adenine $A$, guanine $G$, cytosine $C$ and thymine $T$.

**$\Sigma$:** Our alphabet formed by the mentioned nucleotides is denoted by $\Sigma = \{A, C, G, T\}$

**Cells:** We denote the number of observed single cells by $C$. Please be aware that $C$ is also used for nucleotide base cytosine in a different context.

**Sites $\mathbb{S}$:** We consider a site (or locus) a position in the DNA sequence that refers to the two nucleotides located there, one from each chromatid. The set of sites is denoted by $\mathbb{S}$. Note that for our model a pair of these sites will be used and worked through, therefore we will have elements in $\mathbb{S} \times \mathbb{S}$.

**Reads $R$:** The data we work with are the reads of the cells' DNA sequences. Each cell has more reads or fewer reads at different parts of the DNA sequence, maybe even none in some fragments. All reads together are expressed with $R$, but if we want to specify a read at a given site pair $\boldsymbol{s} \in \mathbb{S} \times \mathbb{S}$ the notation will be $R^{\boldsymbol{s}}_{1:C}$ and for a specific cell $c \in \{1...C\}$, $R^{\boldsymbol{s}}_{c}$.

**Quality scores $Q$:** To each read there is an associated quality score that is denoted by the equivalent notation $Q^{\boldsymbol{s}}_{c}$ for the quality of the reads at the site pair $\boldsymbol{s} \in \mathbb{S} \times \mathbb{S}$ for cell $c \in \{1...C\}$. We assume that they are converted to probabilities.

**Number of reads:** We will use $L^{s}_{c} = |R^{s}_{c}|$, $s \in \mathbb{S}$ to denote the total number of reads for site $s$ for cell $c$.

**Bulk $B$:** The cells gathered as input share the same initial parent: the bulk cell. We assume its cell sequence as known and all cells are a result of its reproduction. Again a specific site pair data at the bulk will be denoted by $B^{\boldsymbol{s}}$.

**Genotype $G$:** During this reproduction of the bulk cell, some mutations may be held at some sites of the DNA sequence, modifying a nucleotide for another. These are called somatic mutations. The genotype of a cell $c \in \{1...C\}$ named and $G^{\boldsymbol{s}}_{c}$ will indicate if the cell is mutated or not at the given site pair $\boldsymbol{s} \in \mathbb{S} \times \mathbb{S}$.

**Fragments $\pi_c$:** We consider the information in fragments $\pi_c = (F_c, \lambda_c), F_c \in \Sigma^2, \lambda \in \mathbb{N}$ which correspond to pairs of nucleotides (one per site in the chosen pair) and to the frequency of that fragment in the reads. For example, we could have ($AC$, 25), ($GG$, 12), ($AT$, 3).

**Identity $\mathcal{I}$:** This function is the identity function for the condition-parameter: if the condition is true the identity is 1 and otherwise 0.

# Chapter 3

# The method

The method used to achieve our goal is composed by two main elements: the model with which we handle the data developed by the Science for Life Laboratory (section 3.1) and the algorithm itself that has been developed explicitly for this thesis (section 3.2). In this chapter, both will be introduced and the used implementation for the algorithm will be presented.

## 3.1 The Site Pair Model

In order to answer our question, it is necessary to first have a clear idea of the data we are handling and how we are using it. The current technology allows the laboratories to read the DNA information from a single cell (instead of a union of many cells, as was done before). This is especially interesting if we are working on mutations of the DNA since neighbour cells may have slightly different sequences that we want to be able to differentiate.

For this purpose, there are two main phases in the process:

1. *Amplification phase*. The DNA sequence of one single cell is not enough to sample the nucleotide sequence because the resolution of the tools is not small enough for it. It must be therefore amplified first, by which we mean that it must be duplicated multiple times to have a larger amount of the same information. To do so, the Multiple Displacement Amplification Method [12] uses a specific enzyme to reproduce the sequence, that goes over the cell's DNA generating its copy. At the end of the process, many copies of the same sequence are available to read.

   These enzymes may make some mistakes when duplicating the sequence and these errors are called amplification errors. The probability for these errors to occur is small enough to assume that at most one amplification error will be

held in the same cell at the same site. It is important to be aware of these errors since they should not be confused with somatic mutations of the cell.

2. *Genome sequencing phase*. Once enough copies are generated, we can start reading the nucleotides sequences. This will result in many segments of the sequence covering the information for a portion of the DNA. Most of the DNA will be covered by several reads when we sequence enough times. In this phase, the read segments will also be fitted in the right order, indicating explicitly what portion of the DNA they cover.

   This phase is also delicate since the provided information could be incomplete. Even if the number of copies is high and the number of reads too, it could be the case that one (or both) of the chromatids did not get any reads at all. Then our reads data set would be missing part of the information. This incident is known as allelic dropout and will be considered in the model to avoid jumping into wrong conclusions.

Once we have the reads of our data prepared, there are many different search strategies to detect somatic mutation effectively to consider. In our case, we will be using the Site Pair Model [2], which allows us to consider a large portion of the data set as the input making the results more accurate to the reality. This model will take into account the possible errors happening on the amplification and sequencing of the DNA sequence driving to a probabilistic framework, as will be explained next.

Note that the bulk information is also known, that is, the original sequence with no mutations or errors is known.

### 3.1.1 Model specification

The Site Pair Model (SPM) is a strategy to identify somatic mutations of DNA sequences by comparing two specific sites of the chromosomes to detect the differences in that pair of sites between all cells. The pair is chosen among all sites $\mathcal{S} > 0$. Thus, let $s, s' \in \mathcal{S}$ such that one of the following scenarios is satisfied:

1. $B^s$ is homozygous, $B^{s'}$ is heterozygous and $\exists R$ read covering both sites.
2. $B^s$ is heterozygous, $B^{s'}$ is homozygous and $\exists R$ read covering both sites.

The selection of the site pairs must ensure that the site pairs are independent to guarantee that the following presented formulas hold. This can be done by looking for pairs of sites separated from each other, in different genome fragments thanks to the assumption $A2$. It is also important to remember that the infinite sites assumption $A1$ implies at most one site is hit by a mutation, either $s$ or $s'$.

Let us take a very small version of a cell lineage tree as an example, as the figure 3.1.1 shows. The root is our bulk cell (ancestral cell) that evolves in time generating

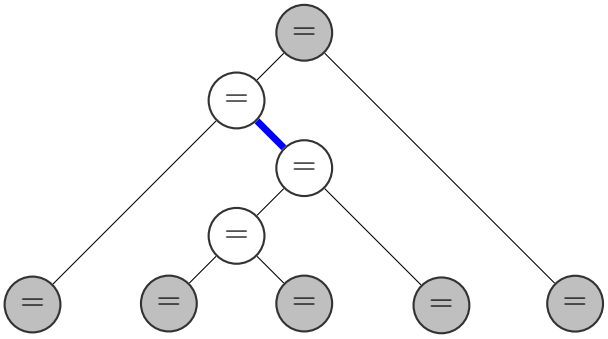


Figure 3.1: Cell lineage tree representation

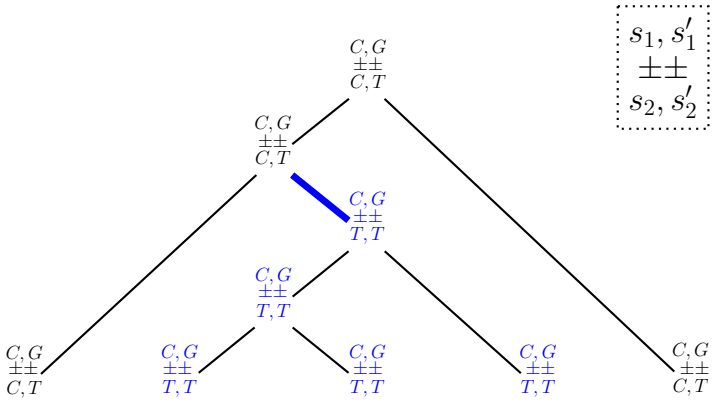



Figure 3.2: Specific nucleotides for the site pair $(s, s')$ per chomatide.

the bottom cells in the leaves. Let $s \in \mathcal{S}$ be a homozygous site harboring a mutation in the blue branch. Note that other mutations could be happening in other sites and branches too.

Once the pair includes the site $s$ with another heterozygous site $s'$, we can see that the data we will read from the cells will be different for the first and last ones than for the ones under the mutation $C \to T$ at the second chromatid. As shown in figure 3.2 they will show fragments $CT$ and $CG$ as in the bulk, while the others will be $TT$ and $CG$ due to the mutation. This allows us to identify subsets of cells that are likely to belong to the same branch of the tree.

Nevertheless it is important to take into account that the reads could contain errors or missing information, as explained before. For example, if the site $s$ was not mutated but an amplification error occurred in the first and second cell we could be tempted to wrongly conclude that they share the same ancestor generating a different tree structure.

In the appendix A the reader may find the graphical representation of the model with all elements that influence the reads for our cells.

### 3.1.2 Relevant concepts

In this section, we will go through the relevant elements in the model for the development of this thesis and formulas from the previous work that will be used further on. This section is very detailed and technical. It aims to share the specific process to the readers interested in reproducing the model.

#### Common mutation types probability

Considering only one mutation in a pair of sites, there are 12 possible mutations: one in each site $s_1, s_2, s_1', s_2'$, and for each site the 3 bases in $\Sigma$ that differ from the bulk $B$. Let $Z$ be a vector of binary variables that indicates which mutation is considered with a 1 and setting the rest to 0 and let $\alpha$ be a vector of 12 Dirichlet positive parameters. We compute the probability for $Z$ through a Dirichlet-Multinomial Distribution with n = 1 trials.

$$P(Z|B,\alpha) = \frac{nBeta(\sum \alpha_k, n)}{\prod_{k:Z_k>0} Z_k Beta(\alpha_k, Z_k)} = \frac{Beta(\sum_{k=1}^{12} \alpha_k, 1)}{Beta(\alpha_z, 1)} \tag{3.1}$$

Generally $\alpha$ will be a vector of 12 ones in this document, giving the same weight to all possible mutation types.

#### Probability of allelic dropouts

An other important variable to take into account is $D_{i,c}$, $i = 1, 2$, which tells us if there is an allelic dropout at one of the cromatids $i$ for a given cell $c$. The probability for this follows a Bernoulli distribution of parameter $p_{ado}$.

$$P(D_{1,c}|p_{ado}) = (p_{ado})^{D_{1,c}}(1 - p_{ado})^{1-D_{c,1}} \tag{3.2}$$

The same probability mass function is used for $D_{2,c}$.

#### Fragments probability

We will consider several cases for the computation of this probability. The representation of the fragments $\pi_c = ((F_{c,1}, l_{c,1}), (F_{c,2}, l_{c,2}), (F_{c,3}, l_{c,3}))$ tells us the frequency of each fragment by defining $l_{c,j} \in 0, ..., |R_c|$ where $j = 1, 2, 3$, and $\sum_j l_{c,j} = |R_c| = L_c$ as a partition of the total number of reads. The tree roots must have the same genotype as the cell's real genotype (either $G_c = Bulk$ or $G_c = Z$, depending on whether we consider the cell mutated or not: $F_{c,1} = G_{c,1}, F_{c,2} = G_{c,2}$), otherwise the probability will be zero. Note that we will define as $F_c = (F_{c,1}, F_{c,2}, F_{c,3})$, $G_c = (G_{c,1}, G_{c,2}, G_{c,3})$ and $\lambda_c = (l_{c,1}, l_{c,2}, l_{c,3})$ to simplify the reading when referring to the three possible genotypes.

The cases are defined as follows:

- **Case 1:** No Allelic Dropout on any allele ($D_{c,1} = 0, D_{c,2} = 0$) ($l_{c,1} > 0, l_{c,2} > 0$).
  - **Case 1.1:** No Amplification Error on both trees ($l_{c,3} = 0$).
    $F_{c,1} = G_{c,1}, F_{c,2} = G_{c,2}, F_{c,3} = \emptyset$
    $l_{c,1} + l_{c,2} + l_{c,3} = L_c, l_{c,1} > 0, l_{c,2} > 0, l_{c,3} = 0$
  - **Case 1.2:** Amplification Error on one of the trees.
    Alternating genotype should be exactly one base different from either first tree's root ($l_{c,3} > 0, d(F_{c,3}||F_{c,1}) = 1$) or the second tree's root ($l_{c,3} > 0, d(F_{c,3}||F_{c,2}) = 1$). In some cases, alternating genotype can be originated from either of the trees.
    $F_{c,1} = G_{c,1}, F_{c,2} = G_{c,2}$
    $F_{c,3} \neq \emptyset, d(F_{c,3}||F_{c,1}) = 1$ and/or $d(F_{c,3}||F_{c,2}) = 1$
    $l_{c,1} + l_{c,2} + l_{c,3} = L_c, l_{c,1} > 0, l_{c,2} > 0, l_{c,3} > 0$
- **Case 2:** No Allelic Dropout on first allele ($D_{c,1} = 0$). Allelic Dropout on second allele ($D_{c,2} = 1$) ($l_{c,1} > 0, l_{c,2} = 0$).
  - **Case 2.1:** No Amplification Error on first tree ($l_{c,3} = 0$).
    $F_{c,1} = G_{c,1}, F_{c,2} = G_{c,2}, F_{c,3} = \emptyset$
    $l_{c,1} + l_{c,2} + l_{c,3} = L_c, l_{c,1} > 0, l_{c,2} = 0, l_{c,3} = 0$
  - **Case 2.2:** Amplification Error on first tree. Alternating genotype should be exactly one base different from first tree's root ($l_{c,3} > 0, d(F_{c,3}||F_{c,1}) = 1$).
    $F_{c,1} = G_{c,1}, F_{c,2} = G_{c,2}, F_{c,3} \neq \emptyset, d(F_{c,3}||F_{c,1}) = 1$
    $l_{c,1} + l_{c,2} + l_{c,3} = L_c, l_{c,1} > 0, l_{c,2} = 0, l_{c,3} > 0$
- **Case 3:** Allelic Dropout on first allele ($D_{c,1} = 1$). No Allelic Dropout on second allele ($D_{c,2} = 0$) ($l_{c,1} = 0, l_{c,2} > 0$).
  - **Case 3.1:** No Amplification Error on second tree ($l_{c,3} = 0$).
    Analogous to Case 2.1, with $l_{c,1} = 0, l_{c,2} > 0$
  - **Case 3.2:** Amplification Error on second tree.
    Analogous to Case 2.2, with $l_{c,1} = 0, l_{c,2} > 0$

Considering this, the computation of the probability of our fragments $F_c$, $\lambda_c$ is defined by conditioning on the variables $B$, $Z$, $G_c$, $D_{c,1}$, $D_{c,2}$, $L_c$ and resulting in the casewise-defined equation 3.3. In the equation $\mathcal{I}$ denotes the indicator function and $p_{ae}$ denotes the probability of allelic dropouts.

### Reads probability

At the end of the day, what we are trying to compute is the probability of the reads that we have obtained, given the characteristics of the model.

We focus first on the information provided by the quality scores on a specific cell $c$. Given a read $i$ of a specific site pair $\boldsymbol{s} = (s_1, s_2)$, we simplify the notation to $R_c^{\boldsymbol{s},i} = r_{s_1}^i r_{s_2}^i$, with $r_{s_1}^i, r_{s_2}^i \in \Sigma$. Then the quality score $Q_c^{\boldsymbol{s},i} = q_{s_1}^i q_{s_2}^i$, with $q_{s_1}^i, q_{s_2}^i \in [0, 1]$ represents the probability of the read being incorrect. This probability will be equally divided by all possible cases of incorrect reads. Then:

$$\begin{aligned} P(r_{s_1}^i = A|C) &= q_{s_1}^i/3 \\ P(r_{s_1}^i = A|G) &= q_{s_1}^i/3 \\ P(r_{s_1}^i = A|T) &= q_{s_1}^i/3 \end{aligned}$$

$$P(r_{s_1}^i = A|A) = 1 - q_{s_1}^i$$

And the same definition is valid for $r_{s_2}^i$ and $q_{c_2}^i$.

Taking a step back, we can express the probability of a site pair read given a specific fragment $F_j$ as specified in equation 3.4. Note that the site pair index is omitted in the notation.

$$P(F_c, \lambda_c | Z, G^s_{1:C}, B, D_{1,1:C}, D_{2,1:C}, P_{ae}) =$$

$$= \begin{cases} \dfrac{1}{L_c - 1}(1 - p_{ae})^{2(L_c-2)}, & \text{if Case 1.1} \\ \dfrac{1}{L_c - 1} p_{ae}(1 - p_{ae})^{2(L_c-2)-1} \dfrac{1}{6} \dfrac{1}{l_{c,3}(l_{c,3}+1)} & \\ \quad \Big( \mathcal{I}\big[d(F_{c,3}||F_{c,1}) = 1\big](l_{c,1} + l_{c,3})! + & \\ \quad + \mathcal{I}\big[d(F_{c,3}||F_{c,2}) = 1\big](l_{c,2} + l_{c,3})! \Big), & \text{if Case 1.2} \\ (1 - p_{ae})^{2(L_c-1)}, & \text{if Case 2.1} \\ p_{ae}(1 - p_{ae})^{2(L_c-1)-1} \dfrac{1}{3} \dfrac{L_c!}{l_{c,3}(l_{c,3}+1)}, & \text{if Case 2.2} \\ (1 - p_{ae})^{2(L_c-1)}, & \text{if Case 3.1} \\ p_{ae}(1 - p_{ae})^{2(L_c-1)-1} \dfrac{1}{3} \dfrac{L_c!}{l_{c,3}(l_{c,3}+1)}, & \text{if Case 3.2} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Equation 3.3. Fragments probability per case.

$$P(R^i, Q^i, F_j) = \begin{cases} (1 - q^i_{c_1})(1 - q^i_{c_2}), & \text{if } r^i_{c_1} = F_{j,1} \text{ and } r^i_{c_2} = F_{j,2} \\ (1 - q^i_{c_1}) \dfrac{q^i_{c_2}}{3}, & \text{if } r^i_{c_1} = F_{j,1} \text{ and } r^i_{c_2} \neq F_{j,2} \\ \dfrac{q^i_{c_1}}{3}(1 - q^i_{c_2}), & \text{if } r^i_{c_1} \neq F_{j,1} \text{ and } r^i_{c_2} = F_{j,2} \\ \dfrac{q^i_{c_1}}{3} \dfrac{q^i_{c_2}}{3}, & \text{if } r^i_{c_1} \neq F_{j,1} \text{ and } r^i_{c_2} \neq F_{j,2} \end{cases} \tag{3.4}$$

Equation 3.4. Reads probability for a given site pair assuming fragment $F_j$.

To compute the probability of the reads according to the given fragments, we have to consider that this probability depends not only on which fragments we have but also

on the frequency of each. Then we can compute this probability recursively as follows since we have to consider all the cases:

*Base cases*: a) There are no fragments left to consider, therefore the probability of the reads is one. b) There are negative frequencies of the fragments, therefore the probability is zero.

*Recursive rule*: The probability of the reads is the sum of considering the first read as the read for each of the fragments, times the probability if the rest of the reads with the frequency of that fragments reduced once.

Writing again $F = (F_1, F_2, F_3)$ and $\lambda = (l_1, l_2, l_3)$:

$$P(i, R^{1:i}|Q^{1:i}, \lambda, F) = \begin{cases} 1, & \text{if } i, l_1, l_2, l_3 = 0 \\ 0, & \text{if } \exists j : l_j < 0, j = 1, 2, 3 \\ Re_1 + Re_2 + Re_3, & \text{otherwise} \end{cases} \quad (3.5)$$

Where $Re_j = P(R^i, Q^i, F_j) * P(i-1, R^{1:i-1}|Q^{1:i-1}, l_1, \dots l_j - 1, \dots l_3, F)$ holds the recursions for $j = 1, 2, 3$.

Then, the probability for all our reads and all elements of the notation is computed with:

$$P(R_c^s|\pi_c, Q_c^s) = P(i, R_c^{s,1:i}|Q_c^{s,1:i}, \lambda_c, F_c), i = L = |R_c^s|$$

This, together with the previous definitions for the probabilities of the mutation types 3.1, allelic dropouts 3.2 and fragments 3.3 permits us to compute the global probability of our reads $R_{1:C}^s$, in the model. The variables $Z$, $D_{c,1}$, $D_{c,2}$, $\pi_{1:C}$ define each a partition of the space, telling us whether something happens or not or which event occurs from a finite and complete list. Therefore, using the *Law of total probability* we can express the probability as a sum of the intersection with all possible scenarios (1). This can be further developed if we condition the mentioned variables taking into account the parameters needed to calculate their probabilities (2). Note that this formula also sums up over all possible mutations so it could have trouble differentiating between mutated cells and amplification errors. An alternative to it is presented in section 6.1 Future Work.

$$
\begin{aligned}
P(R^s_{1:C}|G^s_{1:C}) &\underset{(1)}{=} \sum_{Z} \sum_{D_{1:C},1} \sum_{D_{1:C},2} \sum_{\pi_{1:C}} P(R^s_{1:C}, Z, D_{1:C}, 1, D_{1:C}, 2, \pi_{1:C}|G^s_{1:C}) = \\
&\underset{(2)}{=} \sum_{Z} P(Z|B,\alpha) \prod_{c=1}^{C} \sum_{D_{c,1}} P(D_{c,1}|p_{ado}) \sum_{D_{c,2}} P(D_{c,2}|p_{ado}) * \\
&\quad * \sum_{\pi_c} P(\pi_c|Z, D_{c,1}, D_{c,2}, G^s_c, B, p_{ae}) P(R^s_c|\pi_c, Q^s_c) = \\
&= \prod_{c=1}^{C} \sum_{Z} P(Z|B,\alpha) \sum_{D_{c,1}} P(D_{c,1}|p_{ado}) \sum_{D_{c,2}} P(D_{c,2}|p_{ado}) * \\
&\quad * \sum_{\pi_c} P(\pi_c|Z, D_{c,1}, D_{c,2}, G^s_c, B, p_{ae}) P(R^s_c|\pi_c, Q^s_c)
\end{aligned}
\tag{3.6}
$$

Equation 3.6. Global reads probability per cell and site pair.

## 3.2 Markov chain Monte Carlo

To find an answer to our settled question, we need to figure out how to apply the Markov chain Monte Carlo (MCMC) algorithm to our model. MCMC algorithm is a sampling method that allows sampling from a large class of distributions, and which scales well with the dimensionality of the sample space. This is also the reason why we use the algorithm in our scope: the large amount of data and variables to take into account to generate the tree structure prevent the researcher to use analytical optimizations. There are also other useful methods, like Neighbour-Joining [3], which has been implemented in our laboratory before and to which it would be interesting to compare the results.

This algorithm works by cycles, generating at each cycle a candidate sample (in our scope, a candidate lineage tree) from a prior distribution which depends on the current state. The candidate sample is accepted with a given probability. This probability is given by the ratio between the likelihood of the current sample and the likelihood of the candidate sample. In case it is accepted, it becomes the next current state and we pass to the next cycle. Otherwise, the current state is maintained for the next cycle.

As the number of cycles increases, the distribution from which we extract the samples will tend to the natural distribution for our data and we will be able to sample many lineage trees for this posterior distribution.

The goal is, therefore, to conclude statistical results of the mutations in the cells through the set of evolutionary tree samples from the posterior distribution.

The first and main milestone to achieve this is calculating a lineage tree likelihood based on our model: having as data for each cell the chosen site pair. This will determine how to proceed due to the possibly high computational cost of the likelihood calculations since every cycle executes it at least once.

### 3.2.1 Maximum likelihood for lineage trees

Let us take a deeper look into the likelihood of lineage trees. To calculate the likelihood on the Site Pair Model, we need to adapt the traditional tree likelihood (that considers only one site) so that we can include the information of both sites. Our goal for this step, though, is answering to the same question than the traditional one: how likely are the reads of the cells at each site pair $R_{1:C}^{1:S}$ given a specific cell lineage tree structure $T$?

We want therefore to calculate the following:

$$P(R_{1:C}^{1:S}|T) = \prod_{s=1}^{S} P(R_{1:C}^{s}|T) \tag{3.7}$$

We can consider each pair of sites individually due to the independence of sites. Since we assume the infinite sites model, at most one mutation will affect a given pair of sites and this implies that at most one branch will harbor a mutation.

Let $e$ refer to the edge of the tree with the mutation and $nm$ to the case where no mutations at all take place. Let $E = 2 * C - 2$ be the total number of edges in the tree. Then we can express the probability by adding all possible cases.

$$P(R_{1:C}^{s}|T) = \sum_{e=1}^{E} P(R_{1:C}^{s}, e|T) + P(R_{1:C}^{s}, nm|T) \tag{3.8}$$

Applying conditional probabilities we obtain the following expression for each term of the sum:

$$P(e|T) * P(R_{1:C}^{s}|e, T)$$

And equivalently,

$$P(nm|T) * P(R_{1:C}^{s}|nm, T)$$

From the given information $e$ (or $rm$) and $T$ we can easily obtain the genotype $G_{1:C}^{s}$ which indicates the cells that are affected by the mutation. So, we can simply write the product as $P(e|T) * P(R_{1:C}^{s}|G_{1:C}^{s})$ (analogous for $rm$, where every cell will have a 0 assigned). Combining this with the formulas 3.7 and 3.8 the likelihood stands as:

$$\begin{aligned} P(R_{1:C}^{1:S}|T) = \prod_{s=1}^{S} \Bigg( & \sum_{e=1}^{E} P(e|T) * P(R_{1:C}^{s}|G_{1:C}^{s}) + \\ & + P(nm|T) * P(R_{1:C}^{s}|G_{1:C}^{s}) \Bigg) \end{aligned} \tag{3.9}$$

Thanks to the preparations done in section 3.1.2 we know how to compute $P(R_{1:C}^{s}|G_{1:C}^{s})$ according to the Site Pair Model for any given $G_{1:C}^{s}$ using the equation 3.6. Therefore, the only terms we need to define to be able to calculate the maximum likelihood are $P(nm|T)$ and $P(e|T)$ for $e = 1...E$, the probabilities of the placement of the mutated edge, if any.

One of the most common distributions to model the frequency of an event in time is the Poisson distribution. This seems appropriate to solve our problem if we consider counting the number of mutations that happened in an edge as the events that occur independently. We assume that the probability of a mutation occurring in a given time interval does not vary with time (where time is represented through the length of the branch of the evolutionary tree in our case). W will be note $r$ as the mutation rate per unit length.

Let us consider thus the following parameters:

- $r$ : mutation rate per unit length.
- $l_e$: length of edge $e$. The lengths vary.
- $l$ : sum of all edge lengths $\sum_{e=1}^{E} l_e$.
- $M_e$: mutation status of edge $e$.
- $M$ : total number of mutations $\sum_{e=1}^{E} M_e$.
- $e$ : Euler's constant $2,71828\ldots$ Note that it should not be confused with the edges' subscripts.

Then, the probability of the total number of mutations in the tree follows the distribution given by $P(M = m|r, l_1, l_2, \ldots, l_E) \sim \sum_{e=1}^{E} Poisson(rl_e) \sim Poisson(rl)$:

$$P(M = m|r, l_1, l_2, \ldots, l_E) = \frac{(rl)^m e^{-rl}}{m!} \tag{3.10}$$

Since the sum of independent Poisson random variables is again a Poisson variable with the sum of all parameters as rate.

Consider the Taylor Maclaurin series for the exponential function in equation 3.10 and note that the values $(rl)^2, (rl)^3, \ldots$ are very small.

$$\begin{aligned} e^{-rl} &= \sum_{n=0}^{\infty} \frac{(-rl)^n}{n!} \\ &= 1 - rl + \frac{(rl)^2}{2!} - \frac{(rl)^3}{3!} + \ldots \\ &\approx 1 - rl \end{aligned} \tag{3.11}$$

This allows us to simplify our equation 3.10 to

$$P(M = m|r, l_1, l_2, \ldots, l_E) = \frac{(rl)^m(1 - rl)}{m!} \tag{3.12}$$

Once this is settled we can take a look into the joint distribution $P(M = m, M_1, M_2, \ldots, M_E|r, l_1, l_2, \ldots$ that will indicate us which is the mutated branch. The only case which this joint probability has non-zero value is $m = \sum_{e=1}^{E} M_e$.

$$
\begin{aligned}
P(M = m, M_1, M_2, \ldots, M_E | r, l_1, l_2, \ldots, l_z) = \\
= \mathcal{I}\big[m = \sum_{e=1}^{E} M_e\big] \prod_{e=1}^{E} P(M_e | r, l_e) = \\
= \mathcal{I}\big[m = \sum_{e=1}^{E} M_e\big] \prod_{e=1}^{E} \frac{(rl_e)^{M_e} e^{-rl_e}}{M_e!} = \\
= \mathcal{I}\big[m = \sum_{e=1}^{E} M_e\big] r^m e^{-rl} \prod_{e=1}^{E} \frac{l_e^{M_e}}{M_e!} \approx \\
\approx \mathcal{I}\big[m = \sum_{e=1}^{E} M_e\big] r^m (1 - rl) \prod_{e=1}^{E} \frac{l_e^{M_e}}{M_e!}
\end{aligned}
\tag{3.13}
$$

Therefore the probability of not having any mutation at all (i.e. m = 0 and $\forall e$ $M_e$ = 0 ) can be calculated through the equation 3.13 and simplified to the expression $P(nm|T) = P(M = 0, M_1 = 0, \ldots, M_E = 0 | r, l_1, \ldots, l_E) \approx (1 - rl)$.

Doing the same simplification for the case of one single mutation (i.e. m = 1 and for a given $i$ that indicates the branch harboring the mutation $\forall e, e \neq i$ $M_e$ = 0, $M_i$ = 1) we conclude $P(i|T) = P(M = 1, M_1 = 0, M_i = 1 \ldots, M_E = 0 | r, l_1, \ldots, l_E) \approx rl_i(1 - rl) = rl_i - r^2 l l_i \approx rl_i$.

Going back to the equation 3.9, the likelihood for our model will be calculated using the following formula:

$$
\begin{aligned}
P(R_{1:C}^{1:S}|T) = \prod_{s=1}^{S} \Bigg( \sum_{e=1}^{E} rl_e * P(R_{1:C}^{s} | G_{1:C}^{s}) + \\
+ (1 - rl) * P(R_{1:C}^{s} | G_{1:C}^{s}) \Bigg)
\end{aligned}
\tag{3.14}
$$

Where $G_{1:C}^{s}$ will be determined at each term by the chosen $e$: only the cells in the sub-tree below the chosen $e$ will have mutated genotype.

### 3.2.2 MCMC implementation

In such a problem where the computation of the likelihood is this complex and shall be repeated for each pair of sites times each proposed tree, working for an optimised algorithm is definitely worth it. The difference could reduce hours of computation for the same result. We will therefore first present the optimizations introduced to the formulas above, to understand better how the solution is implemented and presented.

As we will see, implementing this algorithm requires also an extra analysis of the used parameters for the sampling strategy. Each cycle a new tree has to be proposed as a modification of the current state and these modifications will guide the process to different solutions.

### optimization factors

To avoid the repetition of the same calculations we will use dynamic programming, storing the values that we already know, so that on each update we only need to recompute the modified values.

Taking a look into the likelihood final expression 3.14 we can notice two main things. First regarding the internal sum for each edge at which we consider a mutation: the only thing that will change in the read's probability is the genotype of the cells $G^s_{1:C}$. Second regarding the similarities of this genotypes: if we consider two neighbor edges most of the genotypes will probably be the same and can be computed only once.

We use the first fact to reduce the time in the reads' probability computation. Storing the complete reads' probability $P(R^s_{1:C}|G^s_{1:C})$ according to the possible genotypes $G^s_{1:C}$ generates a too big data set. Its size is proportional to the number of possible genotypes, which is exponential in the number of cells: $|\{0,1\}|^C$. Therefore we will consider storing the probability of the reads in the cells $P(R^s_c|G^s_c = 0)$ and $P(R^s_c|G^s_c = 1)$, calculating the value considering the cell mutated and also non-mutated. Then we will be able to calculate our probability by multiplying the corresponding case only, which will reduce the computational cost. Note that the probability for each chosen site pair in each cell will be saved. Every proposed tree will use this common information at every cycle.

The second fact can be used to store sub-trees reads probabilities for either mutated or non-mutated sons, all with the same genotype. When we access another edge that keeps these genotypes the same we will not need to go through the whole tree anymore. This optimizes the computation of the likelihood in each cycle independently.

### Pseudo-code

The pseudo-code is the guidance of the implementation of this Markov chain Monte Carlo algorithm following the previous definitions and can be found at the appendix B.

The first thing to stand out regarding the implementation of the code is that the likelihood, as well as the stored reads probabilities, are calculated with the natural logarithm of themselves due to stability issues of such small probabilities. Otherwise, the product of the probabilities would result in too small values for the computer to process, resulting in big computational errors. This explains the formula for the ratio $R$ for the acceptance decision that is calculated as

$$\begin{aligned} R =& sample\_likelihood/current\_likelihood = \\ =& exp(log(sample\_likelihood/current\_likelihood)) = \\ =& exp(log(sample\_likelihood) - log(current\_likelihood))) = \\ =& exp(sample\_log_likelihood - current\_log\_likelihood) \end{aligned}$$

It does make the likelihood computation more complex since the products become sums and the sums need to be computed by going back to the natural value, adding the exponential form of each and calculating the logarithm of the result again. For the readability of the pseudo-code, this is not detailed.

Another comment regarding efficiency is the computation of the offspring's reads' product: the values are stored in the node so that there is no need to go through the whole tree structure once the value is needed again since the same products are needed in each iteration of the inner for-loop.

Finally, the chosen strategy to make MCMC stop is by the total number of accepted samples. There are different ways of implementing this:

- *Total of accepted samples:* The sampling stops once the total number of accepted samples is reached. By testing different values, the stabilization amount can be found and it is proportional to the data set size. Note that if the acceptance ratio is low this will take longer.

- *Total of generated samples:* The same as the previous case, but considering all samples. Note that if there have been many rejected samples, not enough information will be gathered from the accepted once. The execution time is controlled with this strategy, but not the quality of the resulting information.

- *Likelihood threshold:* Making stop the chain once a specific likelihood value is achieved can be useful for testing the code but not for our purpose: the likelihood is usually unknown and changes a lot from case to case. Besides, we want to keep plenty of samples, not just the one that would reach the threshold.

- *Likelihood stabilization:* To solve the previous option considering the likelihood to stop, the proportion of improvement in it could be regarded. We could stop once there has been no significant improvement in it through enough samples. This requires a larger amount of stored data and computation time since the information from many previous samples is needed. It results in not being worth it compared to the first case.

The first option seems more reasonable to analyze the results of the algorithm, though one could consider using another one for different purposes.

### Sampling strategy

A bit more detailed is the reflection on how to generate a new sample for the Markov chain Monte Carlo algorithm. The strategies considered will determine how the process evolves. The amount of change in the tree has a direct impact on the acceptance ratio of the samples:

- Large modifications in the current tree will result in a low acceptance ratio of the samples. The likelihood of the trees will vary much from a state to the next,

therefore the improvements will be quick when higher, but many samples will be discarded when lower.

- Small modifications in the current tree will result in a high acceptance ratio of the samples. Since the trees will be very similar their likelihoods will be similar too, giving quotient $R$ close to 1 for the acceptance decision.

Ideally, the acceptance ratio should be around 0.4 for an optimal evolution of the process. That is, 40% of the samples are accepted.

In this study, we have applied three different methods to modify the current tree structure for the new cycle.

1. *Adjacent nodes*. This modification consists of interchanging two node locations, the selected one and its parent node location as represented in figure 3.3. This allows subtle modifications in the tree structure, maintaining the global distances.

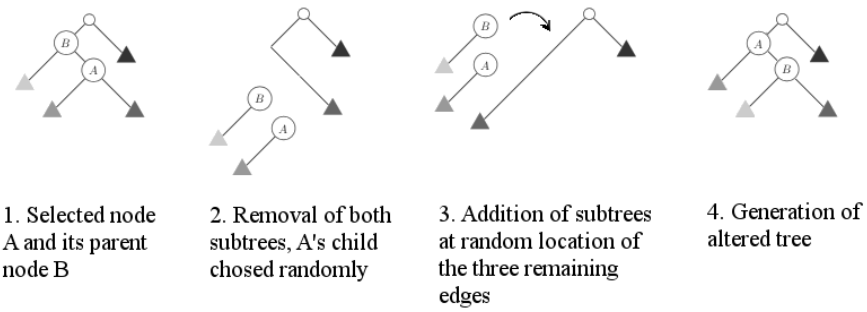



Figure 3.3: Adjacent node modification

2. *Cells location*. All cells are represented at leave nodes and can be interchanged. Thus the tree structure is the same but since the cells are in different locations the result in the likelihood is very different since it has an impact on which cells are now considered mutated or not in each case.

3. *Edges length*. Finally, the lengths of the edges should also be altered and it is not included through the previous methods. For each edge, we will modify them with a given probability in a range between [length / 2, length) or (length, length*2] with the same probability for a shortening of the edge than for an extension of it.

All this modifications of the trees are reversible, that means we can get back to the same tree structure with the same probability with which we left. This quality is necessary for the equilibrium of MCMC that guarantees its well functioning [3].

# Chapter 4

# Results

In this section, the results of the implemented methodologies will be presented. It is important to know that not all tests have been made with the same data sets nor the same computing machine. Smaller synthetic data sets are needed to ensure the proper functionality of the code in a limited time.

The synthetic data generator developed by SciLifeLab provides with the same information of the reads as from a laboratory, besides giving information of the real structure based on the selected sites. The mutations are chosen equally random among all edges, which means that all edges have the same length. Though this length is unknown, so we will only test the shape of the trees when computing the similarities between them, not the pairwise distance among the cells.

## 4.1 Maximum likelihood results

As mentioned at the beginning, a big first step is the computation of the likelihood for the reads given a tree structure. This has been done through the logarithmic likelihood and the mentioned optimizations giving the results shown in the graph 4.1. This first test has been produced with a synthetic data set of 10 cells with a genotype of size 1'000'000 sites of which 190 site pairs are selected with reasonable coverage of reads. Computed for 20 randomly generated samples and 20 modified samples from the real tree structure.

We can observe that the log-likelihood tends to decrease as the symmetric distance between the real simulated tree and the samples increases. This is in our interest since a higher log-likelihood translates to a higher likelihood. It is interesting to see though, that modified samples quick get over 50% difference. Nevertheless, a clear gap can be found between the randomly generated samples in light gray and the modified in dark gray: random samples get clearly lower likelihoods.

In the figure 4.2 a second test for a different data set is represented. Here we

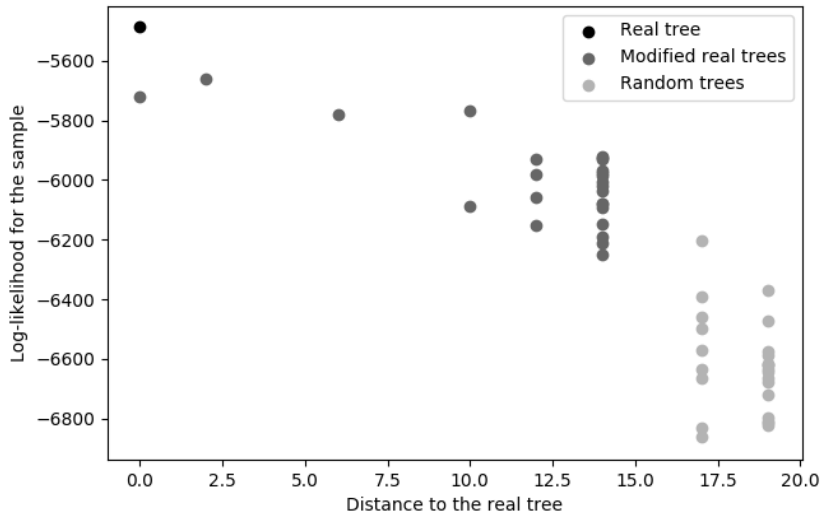



Figure 4.1: Log-likelihood by symmetric difference to the real tree (10 cells)

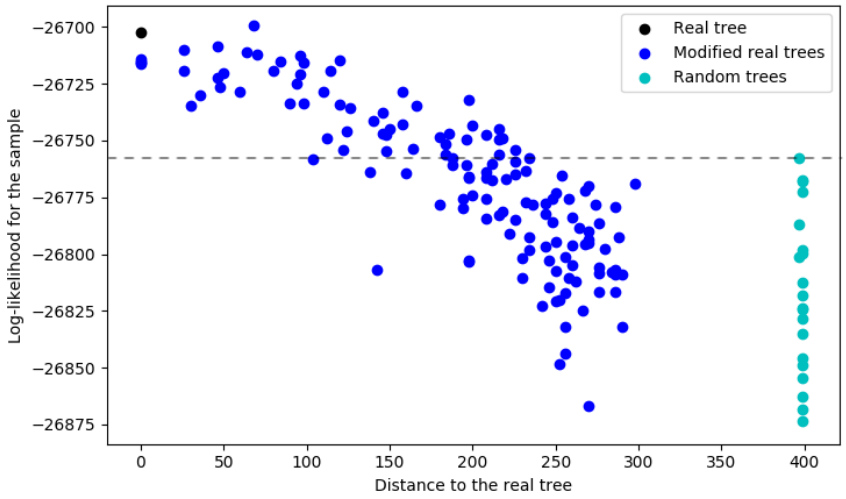



Figure 4.2: Log-likelihood by symmetric difference to the real tree (200 cells)

consider 200 cells, but for each of them a DNA sequence of only 10'000 sites is generated and 80 site pairs considered. The difficulty of this test relies on a higher amount of allelic dropouts and the lack of coverage of reads per cell in the selected sites.

Even with these difficulties the log-likelihood behaves as expected, the closer to the real tree structure, the higher the log-likelihood. Few of the randomly generated trees give high results for the log-likelihood, while slightly modified trees tent to a higher probability. The difference between the two cases is now not as clear as in the previous examples and random samples get quite high in the log-likelihood scale.

A very important aspect of these tests is that even though the results seem satisfactory, the computational cost of them is significantly high: the pre-computation of the probability of the reads per cell, site and genotype taking into account all possible scenarios that need to be computed due to the *Law of total probability* took around twenty hours in a Mac-Book Air with 2 GHz Intel Core i7 processor and 8 GB 1'600 MHz DDR3 memory. The computation of the generation of a new sample together with the log-likelihood had an execution time of 4.9 seconds in mean. This is the portion of the code that needs to be re-computed for each MCMC cycle, the pre-computations will only need to be processed once.

## 4.2 MCMC results

The second results to be tested are the ones provided by the Markov chain Monte Carlo process. At the appendix C the reader can find the detailed parameters for a selection of the tests.

Let's consider the same data sets that the previously considered ones: with 10 cells, medium genome size and reasonable coverage of reads and with 200 cells small genome size and very limited data.

The figure 4.3 show the trace taken by MCMC in the first case. Note that the symmetric difference is in 0 to 1 scale for this section. The trace should be interpreted as follows: for each sample, a gray color is assigned, being darker the closer it is to the posterior distribution (the more cycles of the process, the darker). In the plot we can see how the posterior distribution samples stabilize at 15%-25% of difference towards the the real tree structure, meaning that one or two edges of the tree generate a different partition. This result is quite competent compared to the previous strategy used at SciLifeLab, which was 25% using the neighbour-joining method.

When taking a look into the second scenario we can see that results are not as bright, at figure 4.4. When pushing MCMC to find better tree structures, it manages to do so with clearly distanced trees. All or most of the edges cause different partitions than the real tree structure, so the process can not be trusted to achieve similar trees to the real one at the posterior distribution.

The reason for this is, as we mentioned, the lack of data at the selected sites. Not

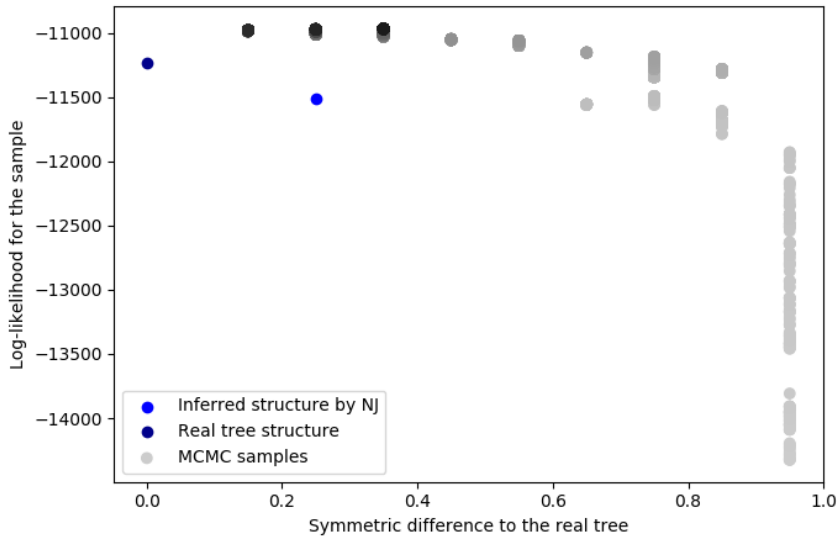



Figure 4.3: MCMC trace for 4000 selected trees

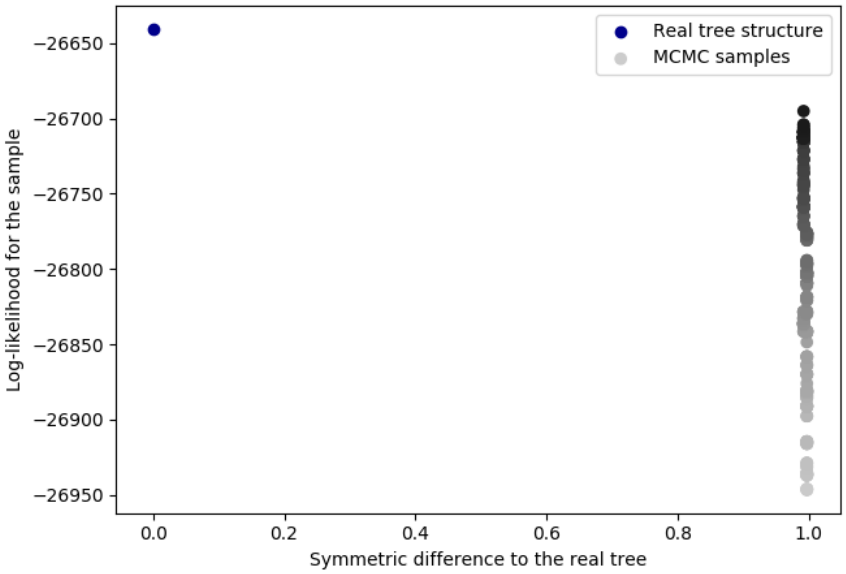



Figure 4.4: MCMC trace for 400 selected trees

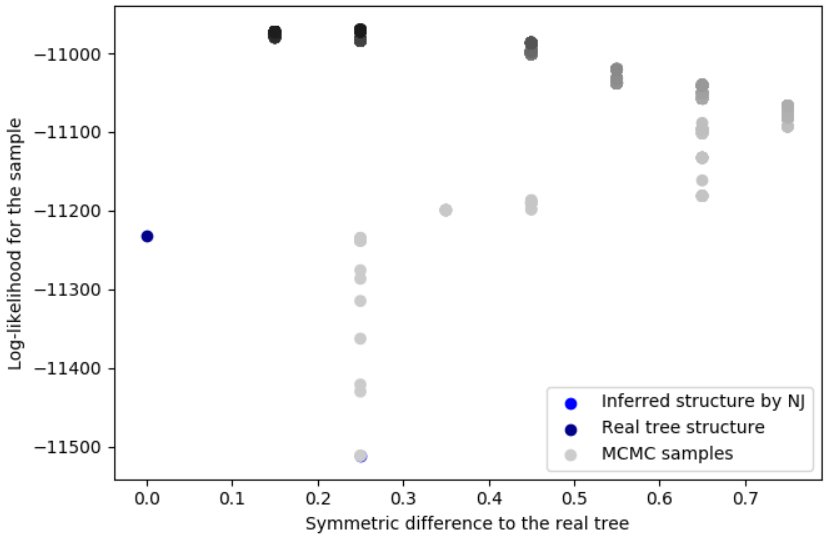



Figure 4.5: MCMC trace for 1000 selected trees with NJ initial state

enough sites are considered (80 site pairs over 10'000 possible sites)[1], and for some of the cells there are no reads that cover the selected sites. For these cells, the location could be any resulting in the same likelihood of the data since there is no data to take into account. In addition, the goal is to find interesting mutations that define our tree in our sites. Very little real mutations are located at any of the site pairs for this sample.

At figure 4.5 we see the representation of the third case: on the 10 cells data set, we compute the Neighbour-Joining result and use this as starting point for our algorithm. One could think that starting so much closer would guarantee closer final results. Nevertheless, when we take a look into the results, the closest we can get to the real tree structure still maintains the same 25% difference. Besides, we can observe how MCMC first moves away from the tree structure and takes about 100 iterations to return to the original difference before improving it.

Note that in the representation the inferred structure by Neighbour-Joining can not be seen because the first grey dot at approx (0.25, -11500) overlaps it. It is exactly where we are starting the sampling.

In terms of results it seems to behave very similarly to the randomized start, but it requires fewer iterations to achieve the same difference rate. This reduces the computation time, which for such slow programs is significantly beneficial.

As mentioned previously, further executions have been made for the presented cases and are presented at appendix C. Different parameters have been chosen and

[1] A minimum reliable quantity of selected site pairs would be around 400 (2*(200-1)) since we are working with 200 cells, which is the amount of edges in the tree.

tested in terms of the number of iterations and the probability of change. As we can observe in the graphs, all executions are different and the results are not every time as close to the real tree structure as expected. It is highly important to consider this variability since it stays out of our hands through the randomized decisions.

# Chapter 5

# Discussion

The obtained results point out some interesting code behaviours that we will discuss in this chapter. We have seen that there are relevant differences in the accuracy of the results according to the parameters chosen.

It is especially relevant to see the impact of limited informative data sets in the results of the algorithm. Not just the chosen algorithm is important but also the type and quantity of data that we handle. The following requirements over the data sets are the ones I consider most important.

1. The data set should have enough DNA reads to cover relevant information for all of the cells.

2. The number of selected sites should be at least double as the number of cells, and the more probability of amplification errors or allelic dropouts the higher it should be. An indicator that can help for realistic probabilities for these errors (for instance 0.6 and 0.00015 respectively) could be an 8% of the genome size.

3. For each cell, most of the selected sites should have reads i.e. it is not enough that a cell has over 100 reads if they all belong to the same fragment of the genome and the rest of sites are not covered at all.

The positive aspect of this limitation is that with proportional amounts to the data that can be obtained in laboratories of real DNA the algorithm does reach accurate tree structures. The obtained result for the second case is coherent taking into account all these issues generated in the data set.

Regarding the algorithm itself, deeper analysis on the parameters for the mutation ratio, probability of amplification error and probability of allelic dropout is currently researched [1]. This will adapt the likelihood computations in the most realistic way according to real data sets.

[1]Research by Hazal Koptagel from Science for Life Laboratory, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

Furthermore, the sampling strategy is an important factor for MCMC, as explained in its section. There is an inverse relationship between the number of samples that we need to generate until the likelihood stabilizes and the probability of change at the modification of the current tree for the new sample. The goal is to find the balance between both to maintain the acceptance ratio around 0.4 [2]. A formula for how low the probabilities of change should be is hard to state due to the multiple parameters of the model in the tree structure and the iterative randomized process. Therefore some empirical tests are required to find the optimal values for these probabilities for each specific data set. The size of the data set and the amount of samples required made it necessary to adjust the probabilities of change to each case.

Nevertheless, the tests made in appendix C.1.3-5 suggest that all three strategies combined generate much better results than on their own. The most useful for our goal in this research is the alteration of adjacent inner nodes. Its tendency is the same as the combined case, but it remains lower than the combined cases tested.

Considering the previous results for this same goal with Neighbour-Joining (NJ) strategy, we can also discuss some interesting issues. Even though the accuracy is higher than the one obtained by NJ in these best cases, the stability of our results is less. There are executions where 10% more of the edges generate the same partition if we remove them, but other executions with equal characteristics may get lower to a 10% less of edges generating the same partition.

This could be solved by executing MCMC multiple times and taking samples of each of the executions, not just from one execution: keeping a mean of the posterior distributions of each execution. This should though be computed and analyzed to conclude if it is reasonable or not.

In any case, this comparison brings the researcher to consider what would happen if we merge both techniques. For this purpose, the third case was made. As mentioned in the results section, the code stabilises very similarly. It even moves away to more different trees before coming back. The reason for it may be that the first moves improve easily enough the likelihood and therefore the restriction over the structure is looser. But when pushing the code to the limit, this structure becomes necessary to keep improving.

[2] Generally accepted optimal ratio. Though, in this study, we see that the best accuracy test was not explicitly related to 0.4 ratio for our tests.

# Chapter 6

# Conclusions

The main answer to the research question is yes, MCMC can be implemented to reconstruct lineage trees from single-cell DNA data following the model developed at SciLifeLab. Getting to closer structures to the real ones will allow biologists to make more reliable conclusions regarding the DNA mutation processes for future medical research.

This study provides a new tool for the reconstruction of lineage trees from single-cell DNA data that shows some improvement towards the previous work on the same model. However, the results still suggest caution when concluding information from these processes, especially when the data set does not provide enough information regarding the reads of the cells. This, together with a proper selection of the site pairs, is crucial for the well functioning of the Markov chain Monte Carlo algorithm.

Regarding the stability of results, that is lower than results obtained by the Neighbour-Joining strategy, it would be interesting to keep improving it. This leaves some questions open towards what could be done for further development. We will present some suggestions in the followings section.

## 6.1 Future work

Some specific ideas that could help to get closer to the real tree structure are proposed here. They are ordered by relevance, subjectively.

1. *Increasing the accuracy of the solutions.* As we mentioned when we introduced the likelihood formula, the current implementation has trouble differentiating cells that have amplification errors. We are adding up the probabilities for all common mutation types instead of considering each mutation type independently. So ideally all the mutated cells should have the same mutation type.

   To do so it is necessary to go back to the equations 3.6 and 3.14. For each

mutated edge that we consider, we will also need to assign which mutation type we are assuming as follows:

$$P(R_{1:C}^{1:S}|T) = \prod_{s=1}^{S} \Bigg( \sum_{Z} P(Z|B,\alpha) \sum_{e=1}^{E} rl_e * P(R_{1:C}^{s}|G_{1:C}^{s}, Z) + \\ +(1-rl) * P(R_{1:C}^{s}|G_{1:C}^{s}, Bulk) \Bigg) \tag{6.1}$$

$$\begin{aligned} &P(R_{1:C}^{s}|G_{1:C}^{s}, Z) = \\ &= \sum_{D_{1:C},1} \sum_{D_{1:C},2} \sum_{\pi_{1:C}} P(R_{1:C}^{s}, D_{1:C}, 1, D_{1:C}, 2, \pi_{1:C}|G_{1:C}^{s}, Z) = \\ &= \prod_{c=1}^{C} \sum_{D_{c,1}} P(D_{c,1}|p_{ado}) \sum_{D_{c,2}} P(D_{c,2}|p_{ado}) * \\ &\qquad * \sum_{\pi_c} P(\pi_c|G_c^s, Z, D_{c,1}, D_{c,2}, B, p_{ae}) P(R_c^s|\pi_c, Q_c^s) \end{aligned} \tag{6.2}$$

This could be the reason why MCMC never obtains tree structures that are closer than 15% difference to the real tree structure. Besides, it would be interesting to see whether it also improves the stability, giving the same quality of results in every execution.

2. *Enhancing the performance in terms of time.* A further implementation would be multiprocessing the likelihood computation since it is a value that needs to be computed at each cycle. For the moment multiprocessing is only included during the preprocessing fragment of the code. One should be careful not to overdo this since all threads will access the same tree values and that could potentially cause a large increase in the synchronization times among the threads. Therefore, too many treads can result in worse execution times.

3. *Gathering more data from further tests.* The current synthetic generator uses equal probabilities for all branches of the tree and does not give a specific value for its length. Since MCMC is also able to adapt the edge lengths, knowing and comparing them would provide more information about how the algorithm works.

   On the other hand, further tests could be done to determine if different parameters get to more stable values. In this thesis, we tested them focusing on obtaining a higher accuracy.

# Bibliography


[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. University of Cambridge. NewYork: Springer Science+Business Media, 2006.

[2] Hazal Koptagel, Seong-Hwan Jun, and Jens Lagergren. "A Probabilistic Framework for Cell Lineage Tree Reconstruction". In: *National Center for Biotechnology Information; US National Library of Medicine, National Institutes of Health* (2018).

[3] Joseph Felsenstein. *Inferring Phylogenies*. University of Washington. Massachusetts: Sinauer Asoci-ates, Inc., 2004.

[4] Daniel Sorensen and Daniel Gianola. *Likelihood, Bayesian, and MCMC Methods in Quantitative Genetics: Statistics for biology and health*. New York: Springer-Verlag New York, Inc., 2002.

[5] Michael A. Lodato et al. "Article PMC4664477: Somatic mutation in single human neurons tracks developmental and transcriptional history". In: *National Center for Biotechnology Information; US National Library of Medicine, National Institutes of Health.* (Oct. 2015).

[6] Charles Gawad, Winston Koh, and Stephen R. Quake. "Single-cell genome sequencing: current state of the science". In: *Nature Reviews Genetics* (Jan. 2016).

[7] Nowell P. C. "The clonal evolution of tumor cell populations". In: *National Center for Biotechnology Information; US National Library of Medicine, National Institutes of Health.* (Oct. 1976).

[8] Mohammed El-Kebir. "SPhyR: tumor phylogeny estimation from single-cell sequencing data under loss and error". In: *National Center for Biotechnology Information; US National Library of Medicine, National Institutes of Health.* (Sept. 2018).

[9] A. W. F. Edwards and L. L. Cavalli-Sforza. “Reconstruction of evolutionary trees”. In: *Phenetic and Phylogenetic Classification* Systematics Association Publication No. 6, London. (1964), pp. 67–76.

[10] Jerzy Neyman. “Molecular studies of evolution: A source of novel statistical problems”. In: *Statistical Decision Theory and Related Topics* Academic Press, New York (1971), pp. 1–27.

[11] Hamim Zafar et al. “Sifit: inferring tumor trees from single-cell sequencing data under finite-sites models”. In: *Genome biology, 18(1):178* (2017).

[12] Frank B. Dean et al. “Comprehensive human genome amplification using multiple displacement amplification”. In: *Proceedings of the National Academy of Sciences of the United States of America* (Apr. 2002).

# Appendix A

# Graphical Model

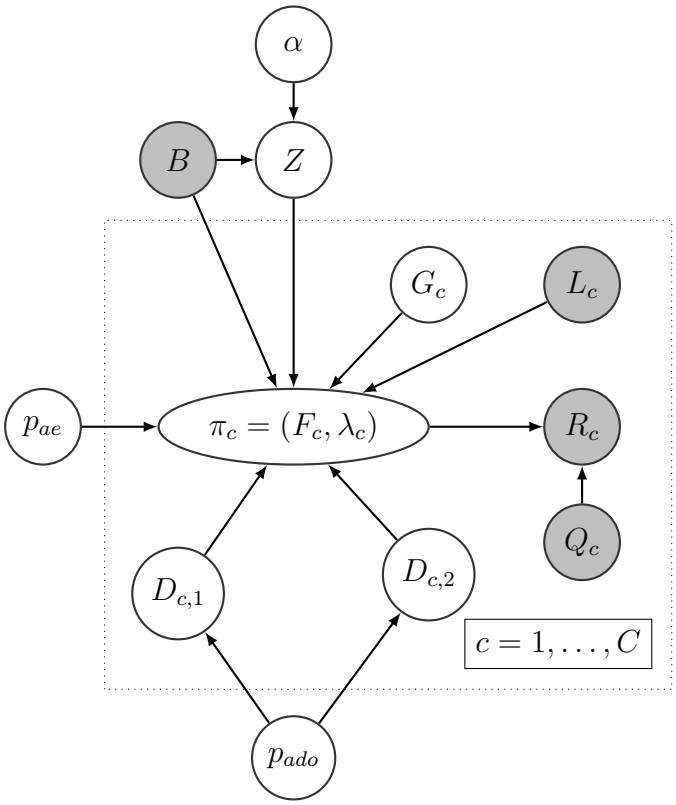



Figure A.1: Graphical Model

The graph in figure A.1 shows the combination of the components taken into account in our model for each site pair $\boldsymbol{s} \in \mathcal{S} \times \mathcal{S}$. For simplicity, the subscript $\boldsymbol{s}$ is omitted in all nodes of the graph.

The observed quantities are filled in grey, referring to the bulk data $B$ (i.e. the ancestral cell data), the number of reads per cell $L_c$, the reads per cell $R_c$ and the quality scores per cell $Q_c$, with $c = 1...C$. The remaining variables are unobserved.

$Z$ denotes the mutation type taken at candidate pair $\boldsymbol{s}$. We assume that $Z$ follows a Dirichlet-Categorical distribution with concentration parameter $\alpha$. This distribution is common for all cells.

$G_c$ are indicator variables for each cell which tell us whether the cell has the analyzed mutation $Z$ ($G_c = 1$) or is still equal to the value taken by the ancestral cell $B$ ($G_c = 0$) in the site pair $\boldsymbol{s}$. In our study we will be always able to calculate this from the proposed tree, looking for the offsprings of the mutated branch and setting only these to 1.

$p_{ado}$ is the probability of allelic dropout, which can take place during the second step of the data processing. It is modeled by the indicator variables $D_{c,i}$, for each cell and for each chromatid $i = 1, 2$:

$$D_{c,i} = \begin{cases} 1 & \text{allelic dropout in chromatid } i \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

$p_{ae}$ is the probability of amplification error, which as we mentioned previously can happen in the first step of the data processing. As we are assuming there won't be more than one error in the pair, the number of different fragments $F_c \in \Sigma^2$ is at most 3: one for the original genotype of each chromatid and one introduced by an amplification error. The representation of the fragments $\pi_c = ((F_{c,1}, l_{c,1}), (F_{c,2}, l_{c,2}), (F_{c,3}, l_{c,3}))$ tells us the proportion of each fragment by defining $l_{c,j} \in 0, ..., |R_c|$ where $j = 1, 2, 3$, and $\sum_j l_{c,j} = |R_c| = L_c$ as a partition of the total amount of reads.

# Appendix B

# Pseudo-code

**Algorithm 1** Markov chain Monte Carlo and relevant called functions

**Input:**
*num_cells*: Number of cells in the data set [int]
*num_iter*: Number of accepted samples in the process [int]
*taxa*: Taxonomy name space for the trees' leaves [taxonomy name space]
*read_probs*: Precomputed probabilities for the reads in the data set [dict]
*ratio*: Mutation ratio of the model [float]
*p_n*: Probability of permutation between adjacent nodes [float]
*p_e*: Probability of modification of an edge length [float]
*p_p*: Probability of permutation between two cells' location [float]
**Output:**
MCMC tree samples

**function** $generate\_new\_tree$(num_cells, taxa):
Generate tree of size 'num_cells' and taxonomy name space 'taxa'.
▷ *Birth and death process*
Initialize node values
**return** *New tree*

---

**Require:** $sample\_tree$(tree, num_cells, p_n, p_e, p_p):
  Duplicate 'tree'
  **for all** node in 'tree' **do**
    **if** random in [0,1) < p_n **then**
      Permute node with its parent node
    **end if**
  **end for**
  **for all** edge in 'tree' **do**
    **if** random in [0,1) < p_e **then**
      Modify edge length to any length between its half and its double
    **end if**
  **end for**
  **for all** leaf in 'tree' **do**
    **if** random in [0,1) < p_p **then**
      Permute leaf node with another random leaf node
    **end if**
  **end for**
  **return** *Modified tree*

**Require:** $compute\_log\_likelihood$:
  log_lh = 0.0
  L = total tree length
  **for all** sites **do**
    Set log_sum_edge to infinite
    **for all** edges **do**
      **if** Seed's edge **then** ▷ *No mutation case*
        Product of precomputed reads for all cells as non-mutated
        Include product*(1-'ratio'*L) to log_sum_edge
      **else** ▷ *Selected edge mutated case*
        Product of all offspring cells as mutated
        Product of the other cells as non-mutated
        Include product*('ratio'*edge length) to log_sum_edge
      **end if**
    **end for**
    log_lh += log_sum_edge
  **end for**
  **return** log_lh

---

**Require:** $main$:
  tree = *generate_new_tree*(num_cells, taxa)
  logLH = *compute_log_likelihood*(tree, read_probs, ratio)
  **while** i < num_iter **do**
    sample = *sample_tree*(tree, num_cells, p_n, p_e, p_p)
    sample_logLH = *compute_log_likelihood*(sample, read_probs, ratio)
    R = exp(sample_logLH - logLH)
    **if** R $\geq$ 1 **or** random in [0,1) < R **then**
      tree = sample
      logLH = sample_logLH
      i += 1
      Store sample when convenient
    **end if**
  **end while**

# Appendix C

# Test results

## C.1 Case 1

| Characteristics of the data set | |
|---|---|
| Number of cells | 10 |
| Length of the genome | 1'000'000 |
| Number of selected site pairs | 390 |

### C.1.1 Execution 1

Execution parameters:

- Probability for interior adjacent nodes modification: 0.02
- Probability for edge lengths modification: 0.05
- Probability for interchange of two cells position: 0.002
- Number of accepted samples required: 4'000
- Initial state: Random tree

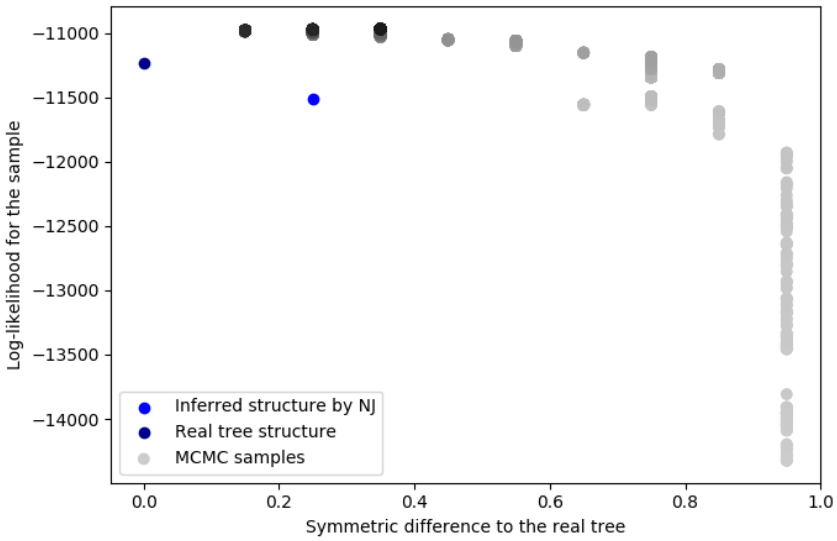



Figure C.1: MCMC trace for 4000 selected trees

Results summary with figure C.1:

- Total execution time for MCMC: 1h 4min 7s (3879.9716029167175s)
- Mean time for new modified sample: 0.111s
- Mean time for logarithmic likelihood computation: 0.225s
- Number of accepted samples: 4’000
- Number of declined samples: 7537
- Acceptance ratio: 0.3467

### C.1.2 Execution 2

Execution parameters:

- Probability for interior adjacent nodes modification: 0.06
- Probability for edge lengths modification: 0.1
- Probability for interchange of two cells position: 0.01
- Number of accepted samples required: 1’000
- Initial state: Random tree

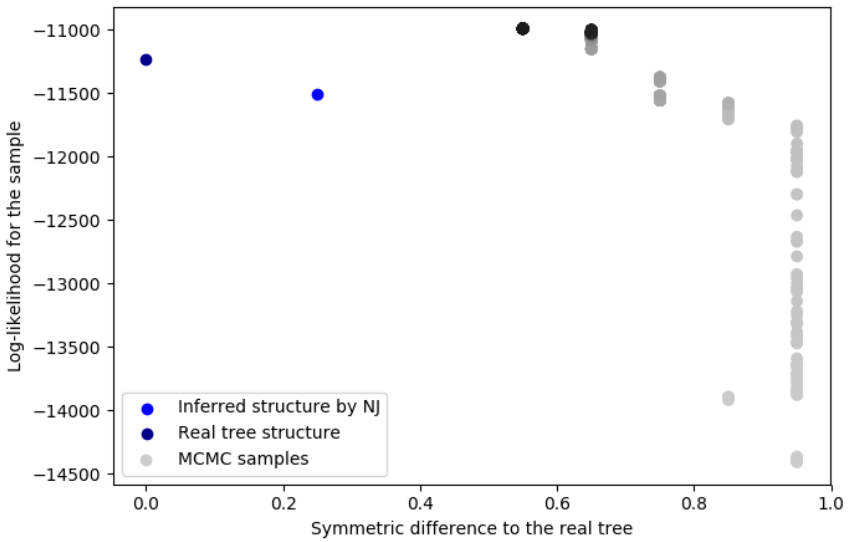



Figure C.2: MCMC trace for 1000 selected trees

Results summary with figure C.2:

- Total execution time for MCMC: 1h 30min 57s (5457.300710916519s)
- Mean time for new modified sample: 0.3858s
- Mean time for logarithmic likelihood computation: 0.7968s
- Number of accepted samples: 1000
- Number of declined samples: 3611
- Acceptance ratio: 0.217

Note that in this test we stop the MCMC samples too early. It is unable to get closer than 50% difference to the real tree structure.

### C.1.3 Execution 3

Execution parameters:

- Probability for interior adjacent nodes modification: 0.08
- Probability for edge lengths modification: 0
- Probability for interchange of two cells position: 0

- Number of accepted samples required: 1’500
- Initial state: Random tree

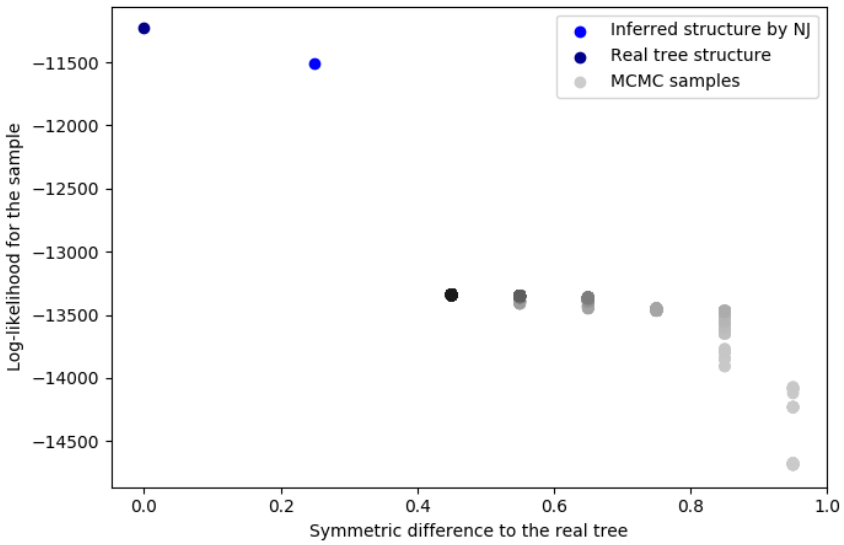



Figure C.3: MCMC trace for 1500 selected trees

Results summary with figure C.3:

- Total execution time for MCMC: 57min 38s (3458.9810271263123s)
- Mean time for new modified sample: 0.412s
- Mean time for logarithmic likelihood computation: 0.896s
- Number of accepted samples: 1500
- Number of declined samples: 1139
- Acceptance ratio: 0.5684

### C.1.4 Execution 4

Execution parameters:

- Probability for interior adjacent nodes modification: 0
- Probability for edge lengths modification: 0.08
- Probability for interchange of two cells position: 0

- Number of accepted samples required: 1’500
- Initial state: Random tree

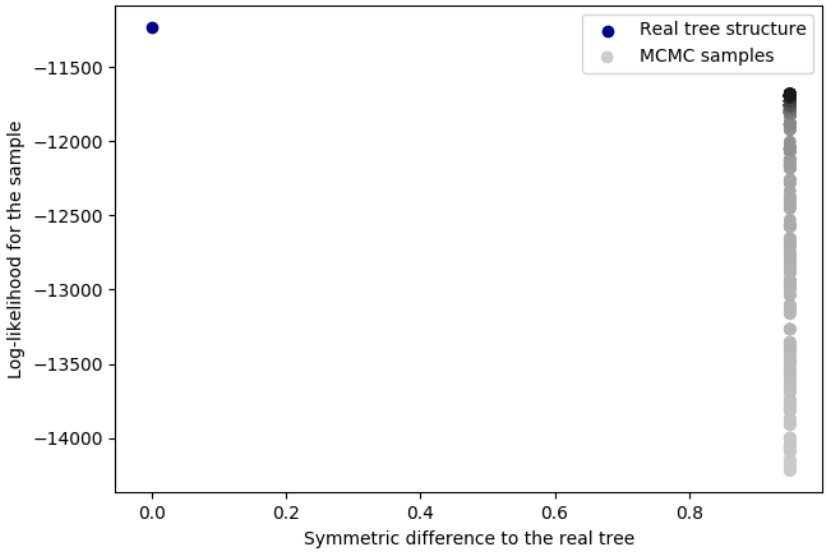



Figure C.4: MCMC trace for 1500 selected trees

Results summary with figure C.4:

- Total execution time for MCMC: 55min 34s (3334.3032190799713s)
- Mean time for new modified sample: 0.401s
- Mean time for logarithmic likelihood computation: 0.708s
- Number of accepted samples: 1500
- Number of declined samples: 1503
- Acceptance ratio: 0.4995

Note that since we only consider the tree structure without the lengths, the tree structure will never get closer to the real one if we only modify edge lengths. Still, we can indeed improve the likelihood through it with MCMC.

### C.1.5 Execution 5

Execution parameters:

- Probability for interior adjacent nodes modification: 0
- Probability for edge lengths modification: 0
- Probability for interchange of two cells position: 0.08
- Number of accepted samples required: 1'500
- Initial state: Random tree

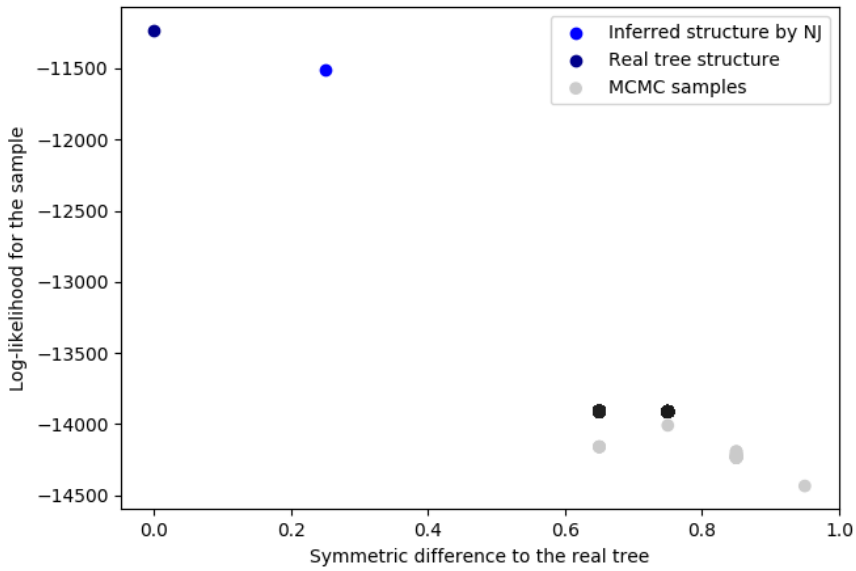



Figure C.5: MCMC trace for 1500 selected trees

Results summary with figure C.5:

- Total execution time for MCMC: 52min 21s (3141.9366812705994s)
- Mean time for new modified sample: 0.387s
- Mean time for logarithmic likelihood computation: 0.702s
- Number of accepted samples: 1500
- Number of declined samples: 1379
- Acceptance ratio: 0.5210

Note that the shape of the structure never changes. Therefore, it is very complicated to achieve high likelihoods by only reordering the leaves of the tree, the cells.

## C.2 Case 2

| Characteristics of the data set | |
|---|---|
| Number of cells | 200 |
| Length of the genome | 10'000 |
| Number of selected site pairs | 80 |

### C.2.1 Execution 1

Execution parameters:

- Probability for interior adjacent nodes modification: 0.005
- Probability for edge lengths modification: 0.01
- Probability for interchange of two cells position: 0.005
- Number of accepted samples required: 200
- Initial state: Random tree

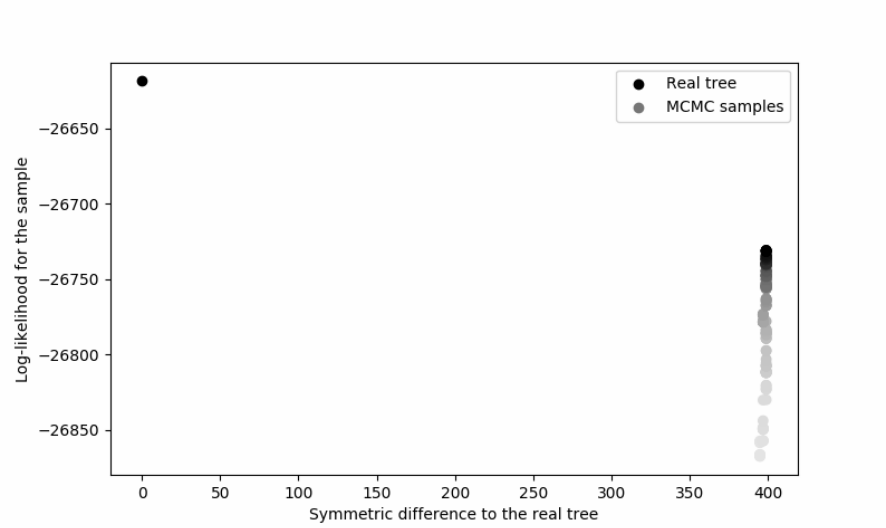



Figure C.6: MCMC trace for 200 selected trees

Results summary with figure C.6:

- Total execution time for MCMC: 30min 3s (1802.8667290210724s)

- Mean time for new modified sample: 1.407s
- Mean time for logarithmic likelihood computation: 4.489s
- Number of accepted samples: 200
- Number of declined samples: 103
- Acceptance ratio: 0.660

### C.2.2 Execution 2

Execution parameters:

- Probability for interior adjacent nodes modification: 0.005
- Probability for edge lengths modification: 0.01
- Probability for interchange of two cells position: 0.005
- Number of accepted samples required: 400
- Initial state: Random tree

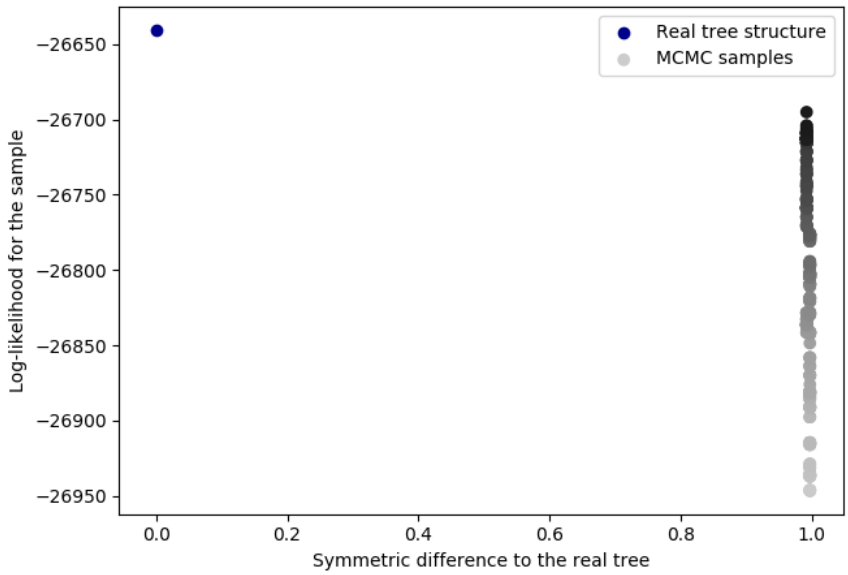



Figure C.7: MCMC trace for 400 selected trees

Results summary with figure C.7:

- Total execution time for MCMC: 1h 45min 30s (6329.886839866638s)
- Mean time for new modified sample: 2.413s
- Mean time for logarithmic likelihood computation: 8.328s
- Number of accepted samples: 400
- Number of declined samples: 185
- Acceptance ratio: 0.684

## C.3 Case 3

| Characteristics of the data set | |
|---|---|
| Number of cells | 10 |
| Length of the genome | 1'000'000 |
| Number of selected site pairs | 390 |

### C.3.1 Execution 1

Execution parameters:

- Probability for interior adjacent nodes modification: 0.06
- Probability for edge lengths modification: 0.1
- Probability for interchange of two cells position: 0.01
- Number of accepted samples required: 1'000
- Initial state: Neighbour-Joining distance method

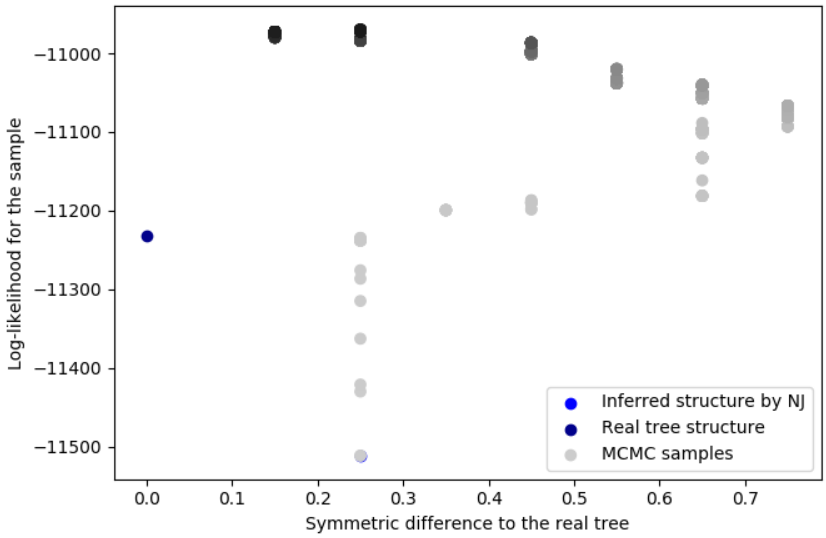



Figure C.8: MCMC trace for 1000 selected trees with NJ initial state

Results summary with figure C.8:

- Total execution time for MCMC: 1h 1min 14s (3674.397349834442s)
- Mean time for new modified sample: 0.111s
- Mean time for logarithmic likelihood computation: 0.218s
- Number of accepted samples: 1000
- Number of declined samples: 10137
- Acceptance ratio: 0.0898

### C.3.2 Execution 2

Execution parameters:

- Probability for interior adjacent nodes modification: 0.07
- Probability for edge lengths modification: 0.1
- Probability for interchange of two cells position: 0.01
- Number of accepted samples required: 1’500
- Initial state: Neighbour-Joining distance method

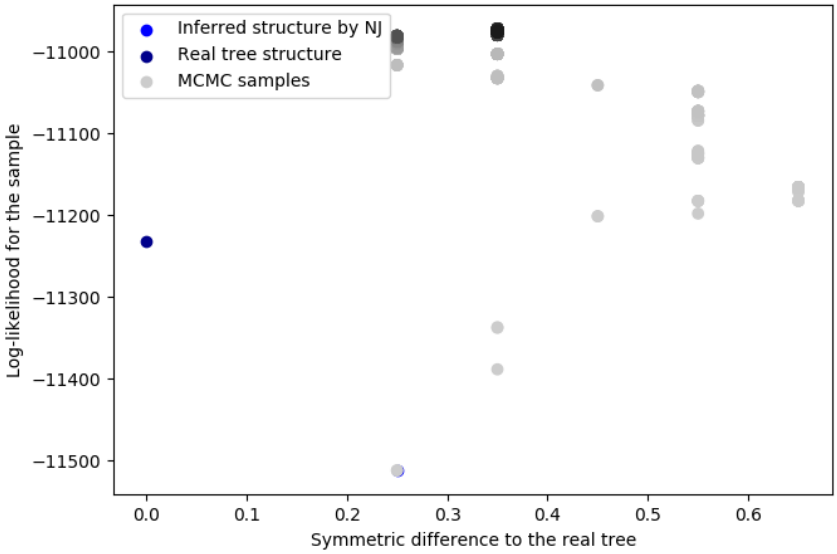



Figure C.9: MCMC trace for 1500 selected trees with NJ initial state

Results summary with figure C.9:

- Total execution time for MCMC: 1h 42min 37s (6157.158034086227s)
- Mean time for new modified sample: 0.110s
- Mean time for logarithmic likelihood computation: 0.218s
- Number of accepted samples: 1500
- Number of declined samples: 17231
- Acceptance ratio: 0.080